

UPS Web Services Sample Code Documentation



Version: 2.00

NOTICE

The use, disclosure, reproduction, modification, transfer, or transmittal of this work for any purpose in any form or by any means without the written permission of United Parcel Service is strictly prohibited.

© 2010 United Parcel Service of America, Inc. All Rights Reserved.

Table of Contents

- 1. Introduction 3
- 2. Axis 2-1.4 Sample Code Naming Convention 3
- 3. JAX- WS 2.1 Sample Code Naming Convention..... 3
- 4. . Net Sample Code Naming Convention 4
- 5. Axis 2-1.4 UPS Web Service Sample Code using ANT build..... 4
- 6. Axis 2-1.4 UPS Web Service Sample Code using Eclipse..... 6
- 7. JAX- WS 2.1 UPS Web Service Sample Code using ANT build 11
- 8. Binding for Ship & Freight Ship Web Service JAX- WS 2.1 Web Service Sample Code 13
- 9. Void Web Service JAX-WS Sample Code..... 14
- 10. . Net(C#) UPS Web Service Sample Code with Microsoft Visual Studio 14
- 11. Build UPS Webservice Sample Code in Perl 15
- 12. Build UPS Webservice Sample Code in PHP 23
- 13. Perl References..... 26
- 14. PHP References..... 27

1. Introduction

IMPORTANT: In order to successfully test transactions, the Client will need to submit transactions containing their own:

- *Access License*
- *UserID*
- *Password*
- *Shipper Account*
- *IZ numbers*

UPS Web Services sample code is developed to assist developers in quick development of UPS Web Service client applications. Developers may use sample code as a reference creating requests and invoking various UPS Web Service. UPS web service client samples are available in Axis 2-1.4, JAX-WS 2.1, .net C#, PERL and PHP technologies.

2. Axis 2-1.4 Sample Code Naming Convention

Sample code Java classes use the following naming convention:

Technology Name + UPS WebService Name +Client

Example 1: **Axis2ShipClient.java**

Axis2 – Technology name

Ship – UPS Web Service Name

Client – Word ‘Client’ is attached to indicate it is client program to invoke UPS web service.

Example 2: **Axis2VoidClient.java**

Axis2 – Technology name

Void – UPS Web Service Name

Client – Word ‘Client’ is attached to indicate it is client program to invoke UPS web service

3. JAX- WS 2.1 Sample Code Naming Convention

Sample code Java classes use following naming convention:

Technology Name + UPS WebService Name +Client

Example 1: **JaxwsShipClient.java**

Jaxws – Technology name

Ship – UPS Web Service Name

Client – Word ‘Client’ is attached to indicate it is client program to invoke UPS web service.

Example 2: **JaxwsVoidClient.java**

Jaxws – Technology name

Void – UPS Web Service Name

Client – Word ‘Client’ is attached to indicate it is client program to invoke UPS web service.

4. . Net Sample Code Naming Convention

Example: **ShipWSClient.cs**

ShipWS - UPS Web Service Name

Client – Word ‘Client’ is attached to indicate it is client program to invoke UPS Web service

Example2 : **VoidWSClient.cs**

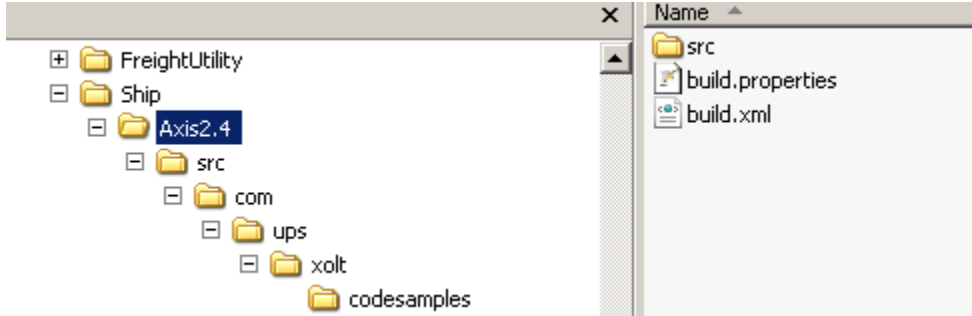
VoidWS - UPS Web Service Name

Client – Word ‘Client’ is attached to indicate it is client program to invoke UPS Web service

5. Axis 2-1.4 UPS Web Service Sample Code using ANT build

- UPS Web Services Axis 2-1.4 sample codes are provided with build.xml. Sample codes have the following directory structure:
- Root directory (name of root directory web service name)
 - src directory (Axis 2-1.4 contains java sample client code)
 - build.xml (ant build script to clean, generate stubs, compile & execute sample code)
 - build.properties (is used by build.xml & client class for reading configurable properties)

Below screenshot shows directory structure of Ship Web Service Java sample code:



Steps for executing build.xml

In order to run build.xml the developer needs to have Apache Ant installed & configured on the machine. Apache Ant is java based build tool. For more details on Apache ant please refer Apache Ant web site: <http://ant.apache.org/>
The response and response status can be found in XOLTResult.xml file. Please refer build.properties for the location of XOLTResult.xml file. The default build target is to run the complete sample program. But for your own purpose, you may select the specific target in the build.xml to run.

Below screen shot shows contains of the build.properties

```
url = CIE URL
accesskey=Your Accesskey
username=Your User Name
password=Your Password
axis2.home = ../../../../lib/Axis2.4
wsdl.home=../../../../SCHEMA-WSDLs
wsdl=Ship.wsdl
mainclass=com.ups.xolt.codesamples.Axis2ShipClient
stubGeneratorclass=org.apache.axis2.wsdl.WSDL2Java
out_file_location =XOLTResult.xml
tool_or_webservice_name = Ship Webservice
src = src
build = build
classes = build/classes
src_dir = src/com/ups/www
```

Below is the response you will get when you open the XOLTResult.xml File

```
<ExecutionAt>Wed Dec 09 15:38:45 IST 2009</ExecutionAt>|
<ToolOrWebServiceName>Ship Web Service</ToolOrWebServiceName>

<ResponseStatus>
  <Code>1</Code>
  <Description>Success</Description>
</ResponseStatus>
```

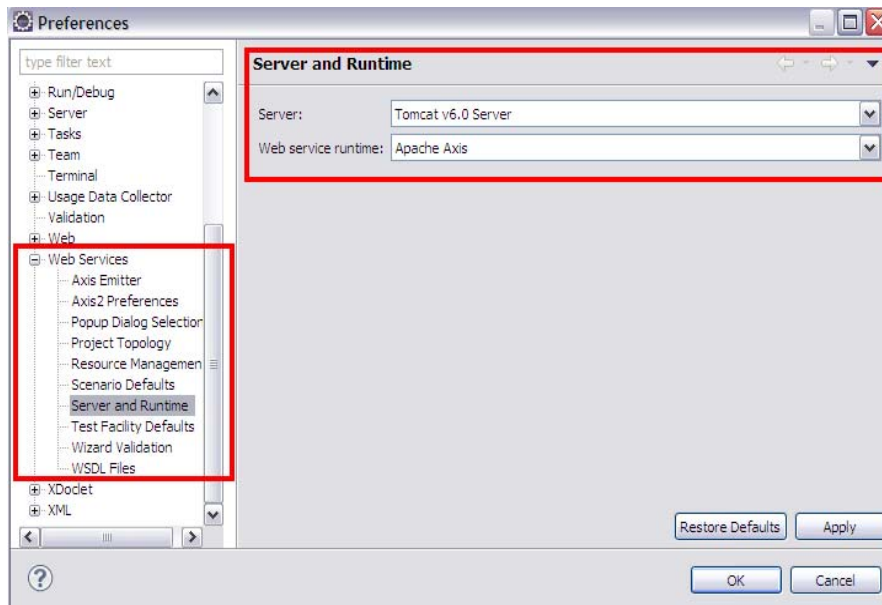
6. Axis 2-1.4 UPS Web Service Sample Code using Eclipse

This section explains UPS Web Services Java sample code by making use of Eclipse IDE with Apache Axis 2-1.4 implementation. Screenshots used in this section are from Eclipse 3.5(Galileo) version. Please refer the following web site to download eclipse IDE for Java EE Developers: <http://www.eclipse.org/downloads/>

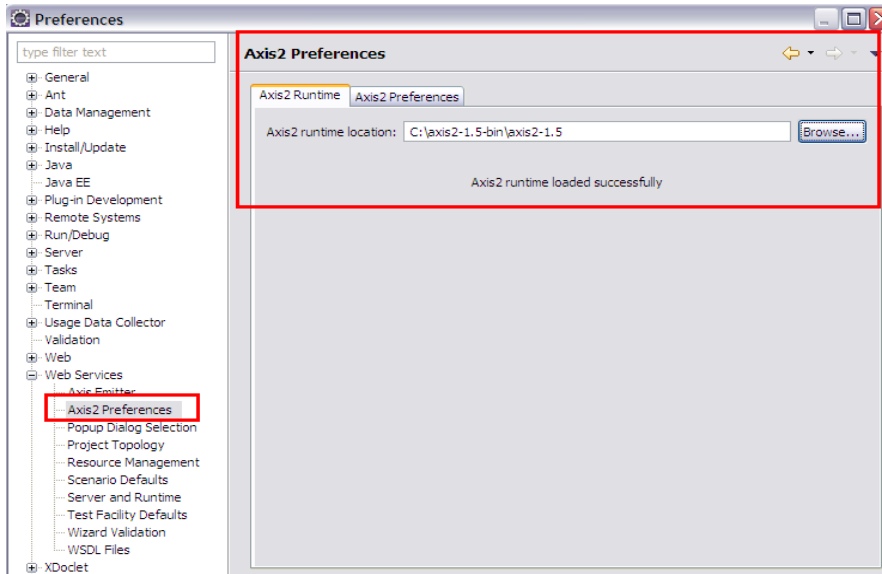
Developers may use any IDE of their choice for development & browsing of sample code, but please note that screenshots & step by step description in this section will be applicable only if one uses Eclipse IDE.

Eclipse configuration steps

- Launch eclipse & navigate to window-> preferences -> Web Services - > Server and Runtime” preferences dialog

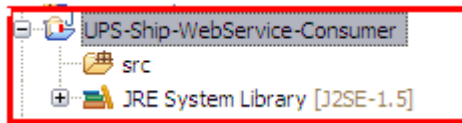


- Choose appropriate Web Service runtime settings.
 - For Apache Axis2, please specify Axis2 runtime location in Axis 2 preferences. This is depicted below
-
- Below is the screen shot

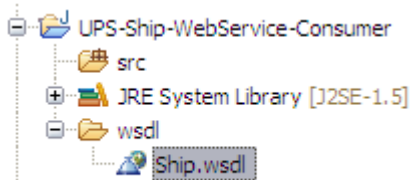


Web Service client project creation steps

- Open Java perspective in eclipse.
- Navigate to menu: File -> New -> Java Project & create a Java project
Example: *UPS-Ship-WebService-Consumer*

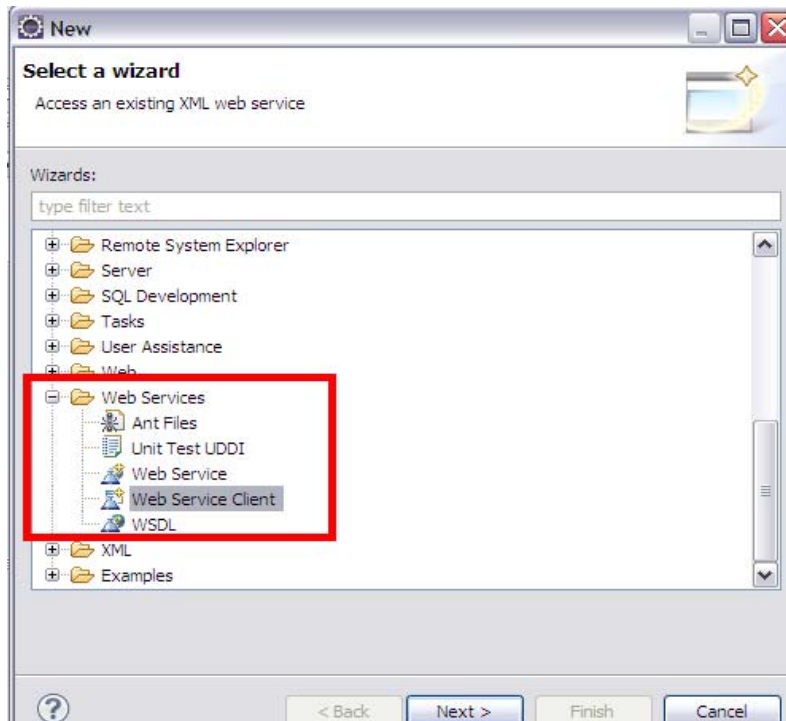


- Create a folder named 'wsdl' in the java project which will contain UPS wsdl file. For example – 'Ship.wsdl'. This is shown below:

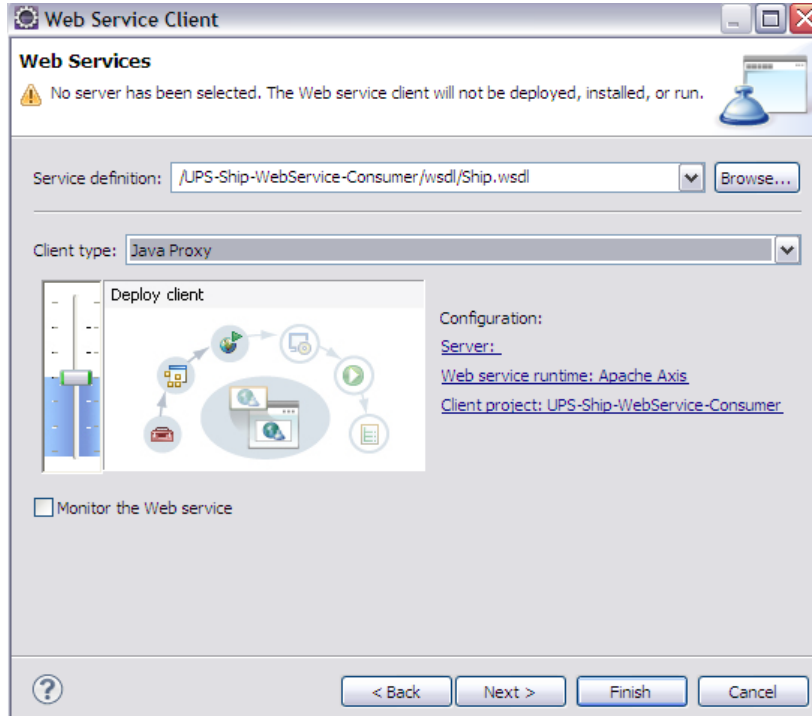


Note: you also need the schemas that are imported by Ship.wsdl in the same directory as Ship.wsdl. Please check out what is imported in the WSDL.

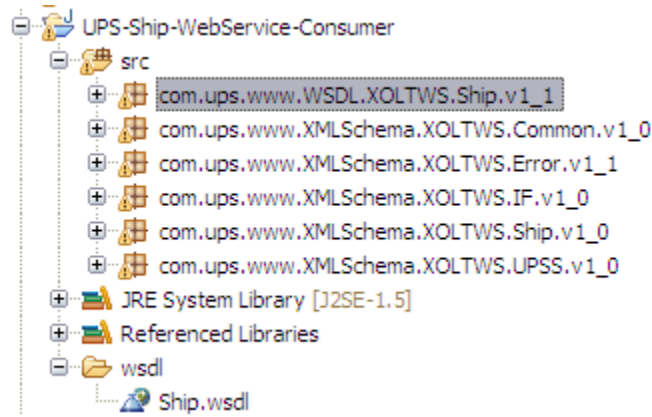
- Select created java project & right click to view context menu. From context menu, select "New -> Other -> Web Services -> Web Service client" as shown below:



- In Web Services client dialog, specify service definition by pointing to wsdl file or wsdl URI.



- Click on Next button to specify output folder for code generation.
- Click on Finish button to generate the stubs.
Below screenshot shows the generated stubs in the java project



- Sample Java client class uses generated stubs to invoke UPS Web Service after creating request & populating the request with right data.

This example makes use of Ship.wsdl Axis 2.4 implementation. Per the naming convention, the name of required client java class will be “AxisShipClient”.

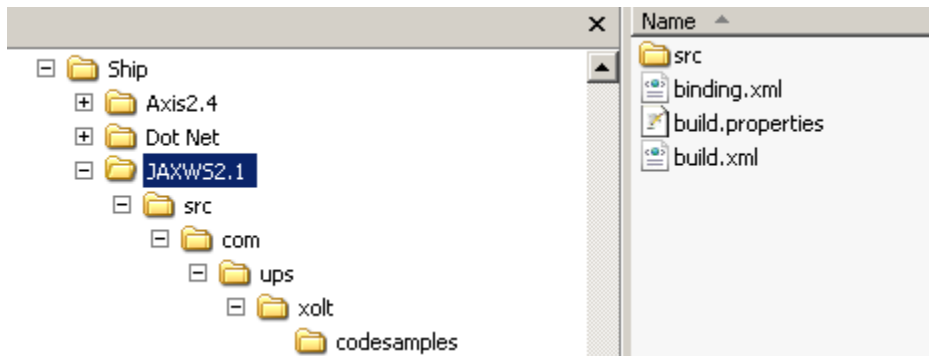
- Copy the sample client Java class to appropriate package structure.
- Copy the build.xml, build.properties in the root directory containing src.
- Developer may update the build.properties file as per the requirements.
- Run the sample Java client class to invoke required web service.
- The response status can be found in XOLTRResult.xml file.
- Please refer to build.properties for the location of XOLTRResult.xml file.

```
axis2.home = ../../../../lib/Axis2.4
wsdl.home=../../../../SCHEMA-WSDLs
wsdl=Ship.wsdl
mainclass=com.ups.xolt.codesamples.Axis2ShipClient
stubGeneratorclass=org.apache.axis2.wsdl.WSDL2Java
out_file_location =XOLTRResult.xml
tool_or_webservice_name = Ship Webservice
```

7. JAX- WS 2.1 UPS Web Service Sample Code using ANT build

- UPS Web Services JAX – WS 2.1 sample codes are provided with build.xml. Sample codes have following directory structure:
- Root directory (name of root directory web service name)
 - src directory (JAX – WS 2.1 contains java sample client code)
 - build.xml (ant build script to clean, generate stubs, compile & execute sample code)
 - build.properties (is used by build.xml & client class for reading configurable properties)
 - Additional binding.xml is needed for ship, freight ship and void web services.

Below screenshot shows directory structure of Ship Web Service Java sample code:



Steps for executing build.xml

In order to run build.xml the developer needs to have Apache Ant installed & configured on the machine. Apache Ant is java based build tool. For more details on Apache ant please refer Apache Ant web site: <http://ant.apache.org/>

The response / response status can be found in XOLTRResult.xml file. Please refer to build.properties for the location of XOLTRResult.xml file. The default build target is to run the complete sample program. But for your own purpose, you may select the specific target in the build.xml to run.

Below screen shot shows contains of the build.properties

```

url = CIE URL
accesskey=Your Accesskey
username=Your User Name
password=Your Password
axis2.home = ../../../../lib/Axis2.4
wsdl.home=../../../../SCHEMA-WSDLs
wsdl=Ship.wsdl
mainclass=com.ups.xolt.codesamples.Axis2ShipClient
stubGeneratorclass=org.apache.axis2.wsdl.WSDL2Java
out_file_location =XOLTResult.xml
tool_or_webservice_name = Ship WebService
src = src
build = build
classes = build/classes
src_dir = src/com/ups/www

```

Below is the sample response you will get when you open the XOLTResult.xml File

```

<ExecutionAt>Wed Dec 09 15:38:45 IST 2009</ExecutionAt>
<ToolOrWebServiceName>Ship Web Service</ToolOrWebServiceName>

<ResponseStatus>
  <Code>1</Code>
  <Description>Success</Description>
</ResponseStatus>

```

Note: For Ship, Void & Freight Ship we have to use binding.xml to get rid of below error while generating stub files.

```

jenolient:
[wsimport] Consider using <depends>/<produces> so that wsimport won't do unnecessary compilation
[wsimport] parsing WSDL...
[wsimport] [ERROR] XPath evaluation of "wsdl:definitions/wsdl:types/xsd:schema[@targetNamespace='http://www.ups.c
[wsimport]   line 10 of file:/C:/projects/DODGE/Release/ERDOCS/shipping/publish/assets/Code%20Samples/Web%20Serv
[wsimport] [ERROR] The package name 'com.ups.xmlschema.xoltws.if.v1' used for this schema is not a valid package
[wsimport]   line 3 of file:/C:/projects/DODGE/Release/ERDOCS/shipping/publish/schema-wsdl/external/IFWS.xsd

```

Refer below section for Ship, Void & Freight Ship web service JAX WS samples only.

8. Binding for Ship & Freight Ship Web Service JAX-WS 2.1 Web Service Sample Code

External binding files are semantically equivalent to embedded binding declarations. We provide the binding.xml to map a name space to an explicit package , so the package name will not have confliction with build-in key word in Java language, such as “if” and “void” in our case.

Below is the sample binding.xml for ship web service.

```
<jxb:bindings version="1.0" xmlns:jxb="http://java.sun.com/xml/ns/jaxb">
  <jxb:bindings xmlns:xs="http://www.w3.org/2001/XMLSchema"
    schemaLocation="wsdl/IFWS.xsd"
    node="/xs:schema">
    <jxb:schemaBindings>
      <jxb:package name="com.ups.IFS"/>
    </jxb:schemaBindings>
  </jxb:bindings>
</jxb:bindings>
```

In this example the child `jaxws:bindings` applies package customization. An XPath expression in the node attribute refers to the root node of the WSDL document, which is `wsdl:definitions` and declares the package `com.ups.IFS` for all the generated Java classes mapped from the WSDL file.

Making use of binding.xml in Build.xml

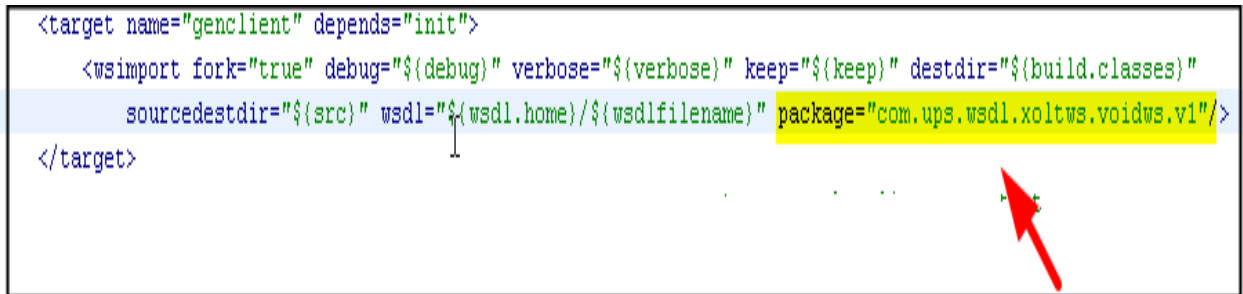
```
<target name="genclients" depends="clean">
  <wsimport fork="true"
    sourcedestdir="${basedir}/src"
    wsdl="${wsdl.home}/${wsdl}"
    binding="${basedir}/binding.xml"/>
</target>
```

9. Void Web Service JAX-WS Sample Code

Build.xml in Void Web Service sample for JAX-WS make use “package” attribute in wsimport element for auto code generation. This ensures that auto generated package names will not have confliction with build-in key word in Java language, such as “if” and “void”.

Below is the sample genclient target indicating use of “package” attribute in wsimport

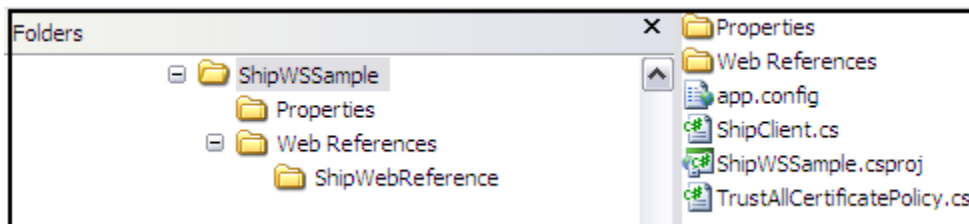
```
<target name="genclient" depends="init">
  <wsimport fork="true" debug="${debug}" verbose="${verbose}" keep="${keep}" destdir="${build.classes}"
    sourcedestdir="${src}" wsdl="${wsdl.home}/${wsdlfilename}" package="com.ups.wsdl.xoltws.voidws.v1"/>
</target>
```

A screenshot of an XML code block. The code is enclosed in a black border. The first line is <target name="genclient" depends="init">. The second line is <wsimport fork="true" debug="\${debug}" verbose="\${verbose}" keep="\${keep}" destdir="\${build.classes}" sourcedestdir="\${src}" wsdl="\${wsdl.home}/\${wsdlfilename}" package="com.ups.wsdl.xoltws.voidws.v1"/>. The third line is </target>. The 'package' attribute value is highlighted in yellow. A red arrow points to the 'package' attribute.

10. .Net(C#) UPS Web Service Sample Code with Microsoft Visual Studio

UPS Web Services .Net sample codes make use of Microsoft Visual Studio 2010.

.Net Sample codes are provided as Visual Studio solution (.sln). All the samples have the directory structure similar to the one below for Ship web service sample:



Open the visual studio solution to browse & execute the sample code. The results can be found on the console. You can modify app.config as you needed to point to the URL which you'd like to test.

11. Build UPS Webservice Sample Code in Perl

- Requirements
- Naming Convention
- Windows Development
- Unix/Linux Development

Requirements

1. Perl 5.8 or above
2. Perl Package Manager
3. XML::Compile::WSDL11 version 2.24 or above
4. XML::LibXML::Simple version 0.91 or above
5. HTTP::Request version 6.02 or above
6. HTTP::Response version 6.02 or above
7. Data::Dumper version 2.131 or above

Naming Convention

Sample Perl code use the following naming convention:

Technology Name + UPS Webservice Name +Client

Example: **XMLCompileWsd11ShipWSCClient.pl**

XMLCompileWsd11 – Technology Name

ShipWS - UPS Web Service Name

Client – Word ‘Client’ is attached to indicate it is client program to invoke UPS Web service

Example2 : **XMLCompileWsd11VoidWSCClient.pl**

XMLCompileWsd11 – Technology Name

VoidWS - UPS Web Service Name

Client – Word ‘Client’ is attached to indicate it is client program to invoke UPS Web service

- UPS Webservice Perl sample code is provided. Perl sample code have the following directory structure:
- Root directory (name of root directory)
 - Perl directory: contains Perl sample client code

Windows Development

11.1 Installing Perl

ActivePerl distribution should be used for Perl development. To download msi executable please refer to the Reference section at the end of this document. A link will be provided to download site.

11.2 Installing Perl modules

We recommend using Perl Package Manager that comes included with ActivePerl distribution. The Perl Package Manager (PPM) can be used to download and install Perl modules automatically.

11.2.1 Firewall/Proxy Configuration

Using PPM requires Internet access to Perl repository sites to download Perl packages. Below in “red” is error message that occurs when PPM cannot connect to Internet.

Downloading ActiveState Package Repository packlist ... failed 407 Proxy Authentication Required

To prevent this error message you must defined a SYSTEM variable inside Environment Variables. Under Environment Variables create a system variable name “http_proxy”. Once system variable and value has been defined restart computer.

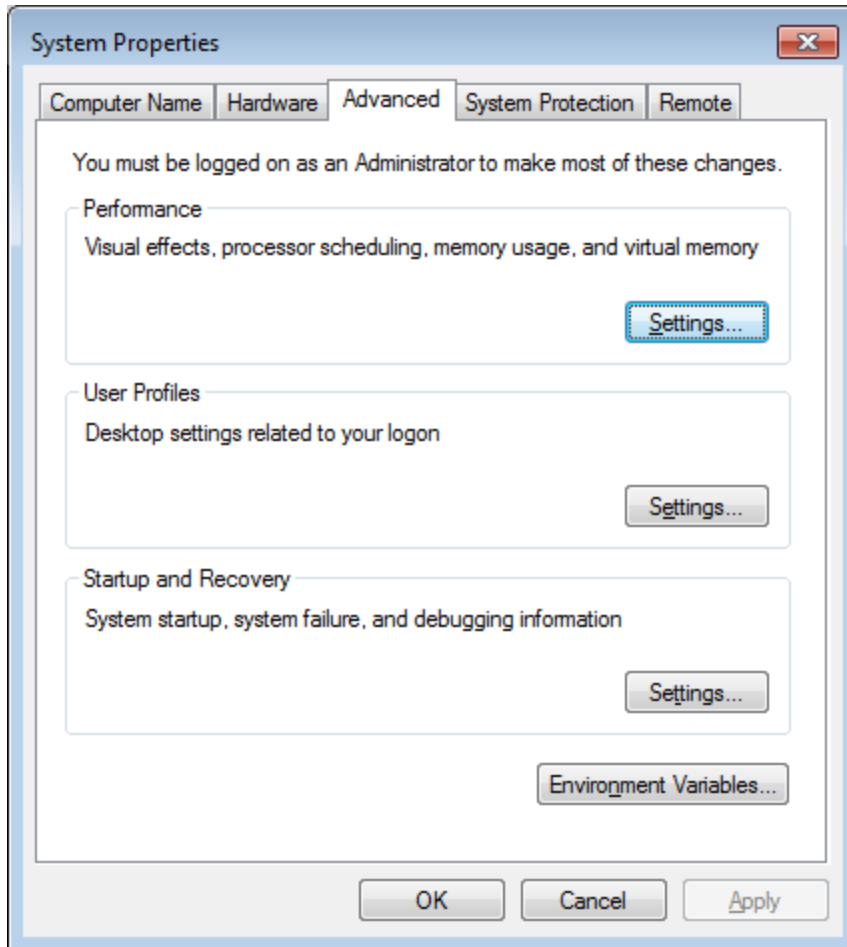


Figure 1: Environment Variables under System Properties

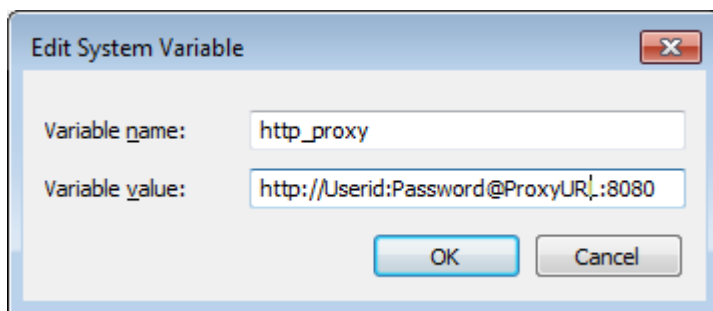


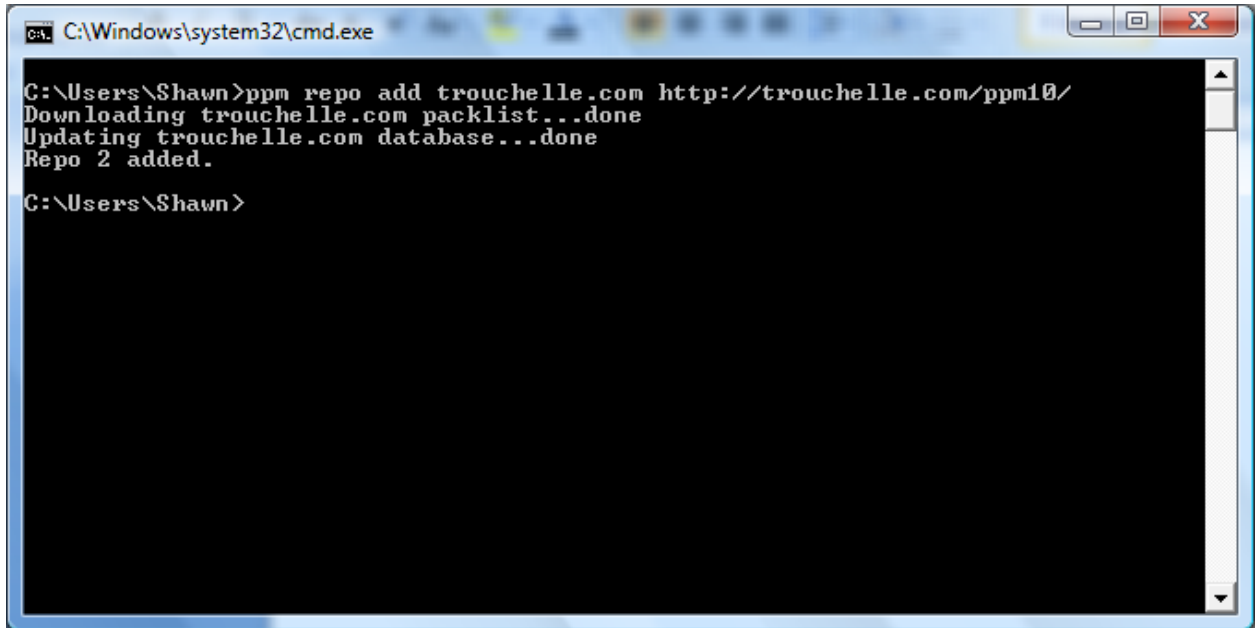
Figure 2: Userid and Password should be the network id and password used to logged onto your network or computer. ProxyURL should be the hostname used to connect to Internet. Please consult your IT administrator to get proxy details.

11.2.2 Perl Package Manager (PPM)

Adding a repository

To add repository, open a command prompt and type the following command

ppm repo add trouchelle.com <http://trouchelle.com/ppm10/>

A screenshot of a Windows command prompt window. The title bar reads 'C:\Windows\system32\cmd.exe'. The command prompt shows the following text:

```
C:\Users\Shawn>ppm repo add trouchelle.com http://trouchelle.com/ppm10/  
Downloading trouchelle.com packlist...done  
Updating trouchelle.com database...done  
Repo 2 added.  
C:\Users\Shawn>
```

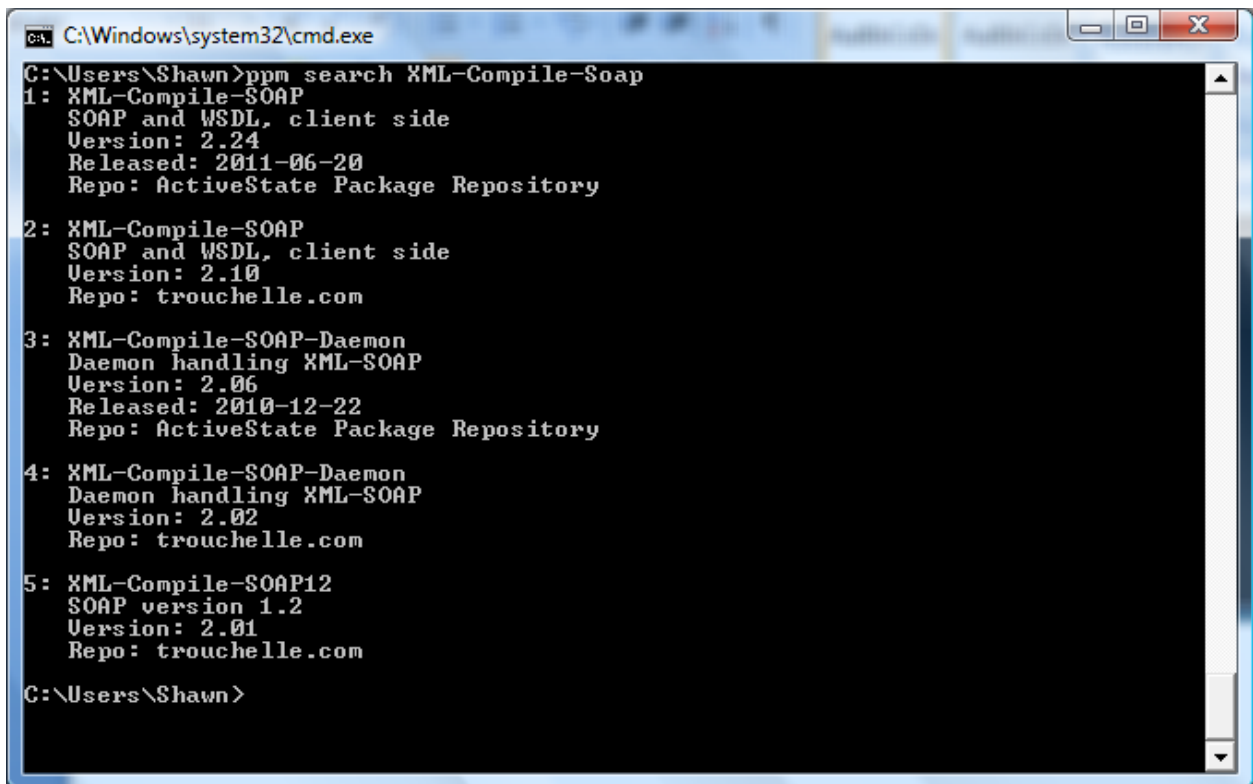
Figure 3: Creating a repository

Installing a Perl module

To install a Perl module for webservices, open a command prompt and execute the following commands in chronological order.

Search command

ppm search XML::Compile::Soap



```
C:\Windows\system32\cmd.exe
C:\Users\Shawn>ppm search XML-Compile-Soap
1: XML-Compile-SOAP
   SOAP and WSDL, client side
   Version: 2.24
   Released: 2011-06-20
   Repo: ActiveState Package Repository
2: XML-Compile-SOAP
   SOAP and WSDL, client side
   Version: 2.10
   Repo: trouchelle.com
3: XML-Compile-SOAP-Daemon
   Daemon handling XML-SOAP
   Version: 2.06
   Released: 2010-12-22
   Repo: ActiveState Package Repository
4: XML-Compile-SOAP-Daemon
   Daemon handling XML-SOAP
   Version: 2.02
   Repo: trouchelle.com
5: XML-Compile-SOAP12
   SOAP version 1.2
   Version: 2.01
   Repo: trouchelle.com
C:\Users\Shawn>
```

Figure 4: Module search for XMLCompileSoap

Install command

ppm install 1

```
C:\Windows\system32\cmd.exe
2: XML-Compile-SOAP
   SOAP and WSDL, client side
   Version: 2.10
   Repo: trouchelle.com

3: XML-Compile-SOAP-Daemon
   Daemon handling XML-SOAP
   Version: 2.06
   Released: 2010-12-22
   Repo: ActiveState Package Repository

4: XML-Compile-SOAP-Daemon
   Daemon handling XML-SOAP
   Version: 2.02
   Repo: trouchelle.com

5: XML-Compile-SOAP12
   SOAP version 1.2
   Version: 2.01
   Repo: trouchelle.com

C:\Users\Shawn>ppm install 1
Downloading XML-Compile-SOAP-2.24...done
Downloading XML-Compile-Cache-0.991...done
Unpacking XML-Compile-SOAP-2.24...done
Unpacking XML-Compile-Cache-0.991...done
Generating HTML for XML-Compile-SOAP-2.24...done
Generating HTML for XML-Compile-Cache-0.991...done
Updating files in site area...done
72 files installed

C:\Users\Shawn>
```

Figure 5: Download XMLCompileSOAP using ActiveState Package Repository site.

We could have chosen to use trouchelle.com repository but it has an older version of XMLCompileSOAP. To download XMLCompileSOAP from trouchelle.com repository, we could have entered “**ppm install 2**” instead of “**ppm install 1**”.

11.2.3 Module dependencies

PPM handles the module dependencies automatically. This reduces the installation time significantly without having to install modules manually.

11.3 Integrate Perl in Eclipse

EPIC is an Eclipse plugin use for Perl development. A link is provided in the Reference section to the main site which will provide a link to the plugin download page and documentation how-to install and integrate the plugin within Eclipse. On the download page choose the latest EPIC plugin.

11.4 Build UPS WebService in Eclipse using XMLCompileWSDL11

11.4.1 Create a new Perl project in Eclipse

11.4.2 Import XMLCompileWsd11ShipWSClient.pl to project

11.4.3 Define the values for the following variables accordingly

- access license, userid and password
- endpoint url
- wsdl file
- operation name
- schema directory location
- outputFileName – response will get save here

11.4.4 Run XMLCompileWsd11ShipWSClient.pl

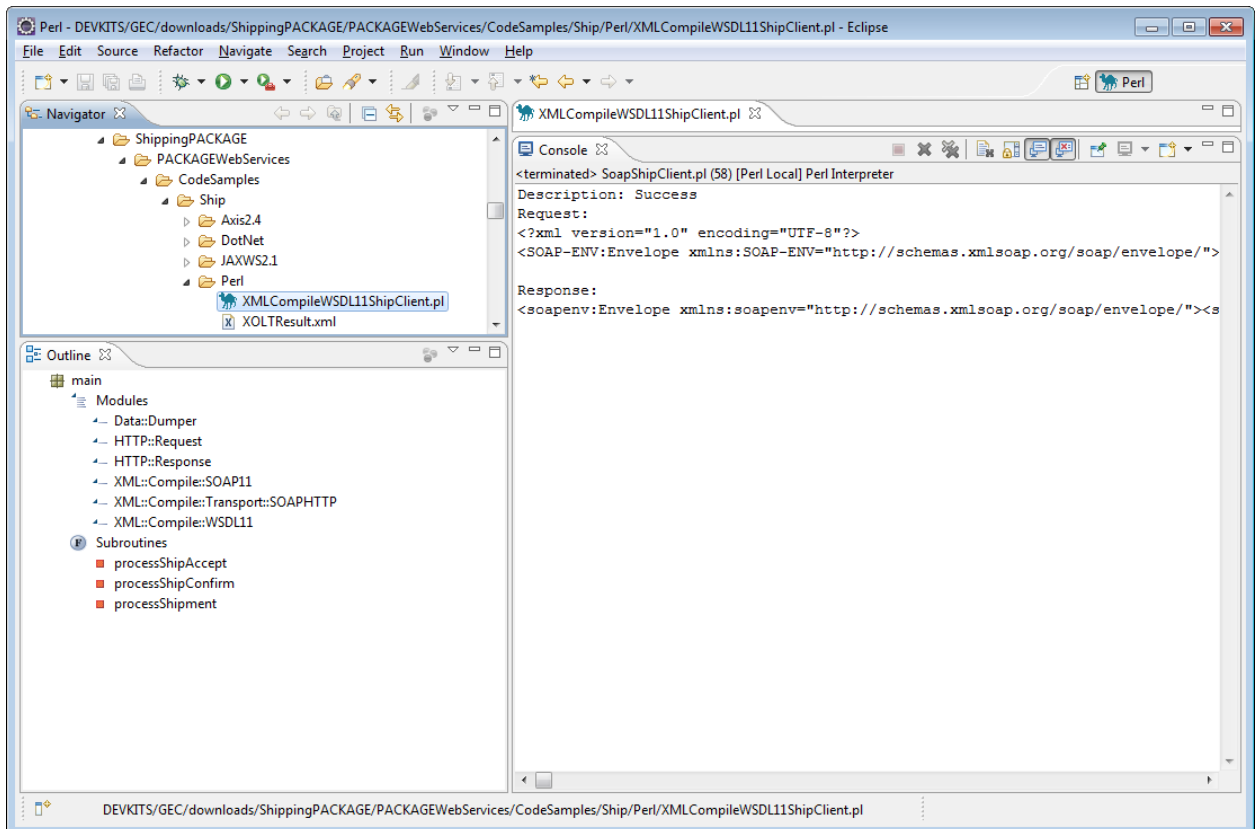


Figure 6: This diagram shows Ship soap request and response message.

```
Perl - DEVKITS/GEC/downloads/ShippingPACKAGE/PACKAGEWebServices/CodeSamples/Ship/Perl/XOLTRestult.xml - Eclipse
File Edit Source Navigate Search Project Run Window Help

XMLCompileWSDL11ShipClient.pl X XOLTRestult.xml

Request:
POST http://[redacted]/xoltws_ship/Ship HTTP/1.1
User-Agent: libwww-perl/5.834
Content-Length: 3836
Content-Type: text/xml; charset="utf-8"
SOAPAction: "http://onlinetools.ups.com/webservices/ShipBinding/v1.0"
X-LWP-Version: 5.834
X-XML-Compile-Cache-Version: 0.991
X-XML-Compile-SOAP-Version: 2.24
X-XML-Compile-Version: 1.22
X-XML-LibXML-Version: 1.70

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Header><ups:UPSSecurity xmlns:ups

Response:
HTTP/1.1 200 OK
Date: Thu, 17 Nov 2011 15:43:46 GMT
Pragma: no-cache
Content-Type: text/xml
Client-Date: Thu, 17 Nov 2011 15:43:43 GMT
Client-Peer: [redacted]
Client-Response-Num: 1
Client-Transfer-Encoding: chunked
X-Powered-By: Servlet/2.5 JSP/2.1

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"><soapenv:Header/><soapenv:Body><ship:ShipmentF
```

Figure 7: This diagram shows a detailed description of the soap request and response in the XOLTRestult.xml

UNIX/Linux Development

11.5 Installing Perl

Depending on the setup of your UNIX/Linux operating system, Perl should be already installed on this machine. However, if Perl is not installed on this machine, we strongly recommend consulting with your IT administrator to have this configured.

Note* Must be logged into the system at root level to install Perl.

11.6 Installing Perl modules

There are various tools that can be used like PPM to install Perl modules. In the Reference section we provided a link for information on various tools that can be used for installing Perl modules on UNIX/Linux system.

11.6.1 Firewall/Proxy Configuration

Please consult with your IT administrator to determine if your UNIX/Linux system is behind a firewall that will prevent downloading Perl modules.

11.6.2 Dependencies

To determine what dependencies are needed for UNIX/Linux environment, we strongly recommend consulting with your IT administrator to gather this information before installing Perl and Perl modules.

Note* Must be logged into the system at root level to configure firewall and install Perl module dependencies.

11.7 Build UPS WebService in Eclipse using XMLCompileWSDL11

11.7.1 Open XMLCompileWsd11ShipWSCClient.pl in VI editor

11.7.2 Define the values for the following variables accordingly

- access license, userid and password
- endpoint url
- wsdl file
- operation name
- schema directory location
- outputFileName – response will get save here

11.7.3 Run XMLCompileWsd11ShipWSCClient.pl using the following commands

```
perl XMLCompileWsd11ShipWSCClient.pl
```

12. Build UPS WebService Sample Code in PHP

- Requirements
- Naming Convention
- UNIX/Linux Development

Requirements

1. PHP 5.3 or above
2. Soap extension

Note* PHP highly recommends building PHP extensions on a UNIX/Linux environments preferably.

Naming Convention

Sample PHP code use the following naming convention:

Technology Name + UPS WebService Name +Client

Example: **SoapShipWSClient.php**

SoapClient – Technology Name

ShipWS - UPS Web Service Name

Client – Word ‘Client’ is attached to indicate it is client program to invoke UPS Web service

Example2 : **SoapVoidWSClient.php**

SoapClient – Technology Name

VoidWS - UPS Web Service Name

Client – Word ‘Client’ is attached to indicate it is client program to invoke UPS Web service

- UPS WebService PHP sample code is provided. PHP sample code have the following directory structure:
- Root directory (name of root directory)
 - PHP directory: contains PHP sample client code

UNIX/Linux Development

12.1 Installing PHP

Please consult with your IT administrator about the UNIX/Linux binaries needed for PHP installation.

Note* Must be logged into the system at root level to install PHP.

12.2 Installing Soap Extension

Please consult with your IT administrator about installing PHP extensions on your UNIX/Linux environment. Before installing soap extension please check the “ext” directory under PHP folder. For example, “/php5-3-6/ext” and verify if soap directory is present. Inside this directory is a hidden folder called “.libs” and it will contain a soap.so file. If this file in this directory exists then soap extension has been built and may need to be enabled. If this file does not exist or the directory itself is not present then soap extension may need to be built.

12.2.1 Build Soap

A link to php.net site is provided in the Reference section that can help with building and installing PHP extensions on UNIX/Linux system. We recommend

using `phpize` command for simplicity. Please consult with your IT administrator for this process. After build, the extension will be installed to a specific directory. This directory is mention after the build finishes. This extension contains the SoapClient program and has to be enabled for PHP to use.

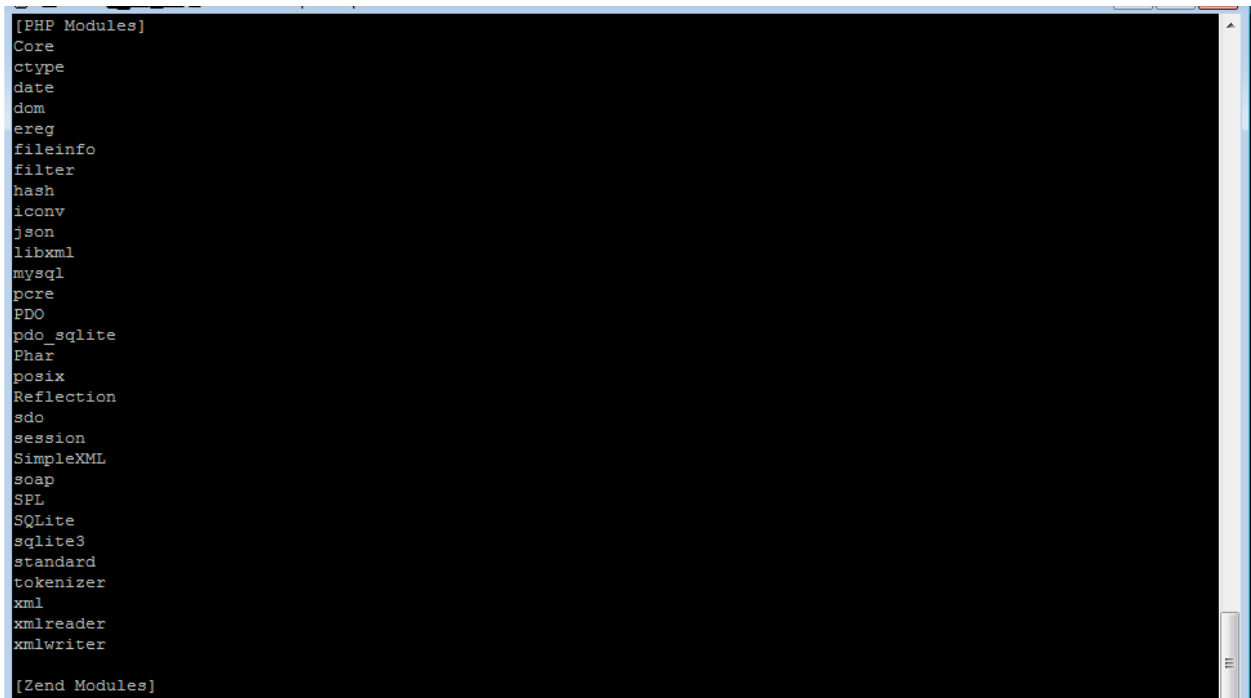
Note* Must be logged into the system at root level to build SoapClient extension.

12.2.2 Configure PHP.ini

Once the soap extension has been built successfully, locate and open `php.ini` in VI editor. Enable the soap extension by adding the file location where the `soap.so` extension is installed.

To verify if the soap extension was enabled successfully run the following command

`php -m`



```
[PHP Modules]
Core
ctype
date
dom
ereg
fileinfo
filter
hash
iconv
json
libxml
mysql
pcre
PDO
pdo_sqlite
Phar
posix
Reflection
sdo
session
SimpleXML
soap
SPL
SQLite
sqlite3
standard
tokenizer
xml
xmlreader
xmlwriter

[Zend Modules]
```

Figure 1: This diagram shows soap extension is enabled

12.2.3 Dependencies

To determine what dependencies are needed for UNIX/Linux environment, we strongly recommend consulting with your IT administrator to gather this information before installing PHP and Soap extension.

12.3. Build UPS Webservice in VI using SoapClient

12.3.1 Open SoapShipWSClient.php in VI editor

12.3.2 Define the values for the following variables accordingly

- access license, userid and password
- endpoint url
- wsdl
- operation name
- outputFileName – response will get save here

12.3.3 Save file

12.3.4 Run SoapShipWSClient.php using the following command

php SoapShipWSClient.php

13. Perl References

Download site for ActivePerl distribution:

<http://www.activestate.com/activeperl/downloads>

Tools use for installing Perl modules: <http://www.cpan.org/modules/INSTALL.html>

XML::Compile::WSDL11 module reference site: <http://search.cpan.org/~markov/XML-Compile-SOAP-2.24/lib/XML/Compile/WSDL11.pod>

XML::Compile::Schema module reference site: <http://search.cpan.org/~markov/XML-Compile-1.22/lib/XML/Compile/Schema.pod>

EPIC plugin tutorial site: <http://www.epic-ide.org/>

EPIC plugin download site: <http://sourceforge.net/projects/e-p-i-c/files/e-p-i-c/>

Perl repository site to download and install XML::Pastor:

<http://trouchelle.com/perl/ppmreview.pl>

Installing Perl modules (Manual Process) - <http://www.thegeekstuff.com/2008/09/how-to-install-perl-modules-manually-and-using-cpan-command/>

14. PHP References

Build and compile PHP extensions: <http://www.php.net/manual/en/install.pecl.phpize.php>

SoapClient Guide: <http://us.php.net/manual/en/class.soapclient.php>